Übung Embedded and Pervasive Systems	Institut für Pervasive Computing
SS 2009, Übung 3	Univ.-Prof. Dr. Alois Ferscha
Elektr. Abgabe: Bis Di, 21. April 2009, 12.00 Uhr	Dipl.-Ing. Dr. Andreas Riener

Zuname:				Tutor:	
Vorname:				Übungsgruppe:	
Matr. Nr.:		SKZ:		Punkte (max. 24)	

Signalverarbeitung: Morsecode (14+10=24 Punkte)

Die **großen Vorteile** bei der Übertragung von Informationen mittels Morsecode sind:

- Durch den einfachen Signalaufbau werden kaum Anforderungen an die Hardware zum Senden bzw. Empfangen gestellt.
- Der Code kann auch bei sehr ungünstigem Signal-Rausch-Verhältnis noch entschlüsselt werden.
- Die Übertragung geht äußert sparsam mit Bandbreite um.

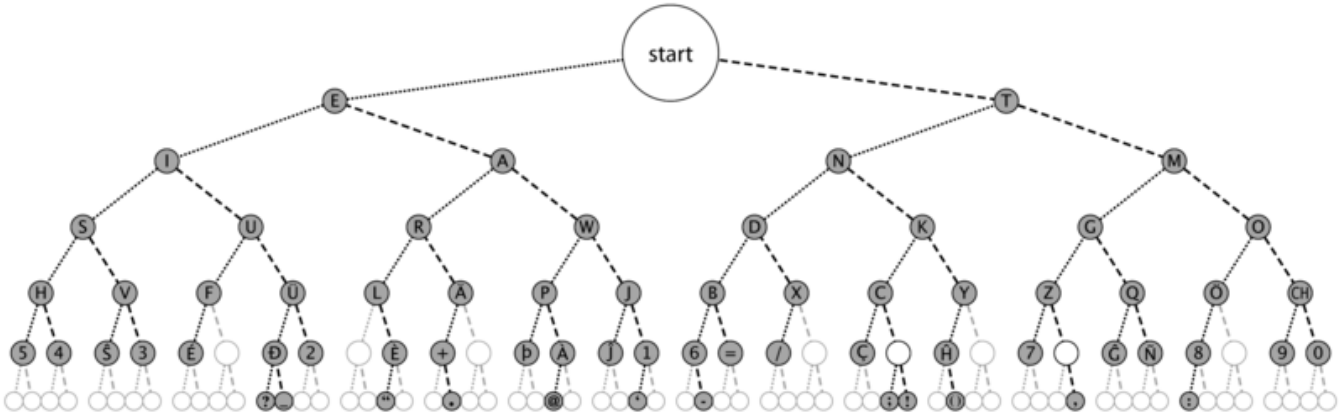
Aus all diesen Gründen hat sich der Morsecode in der Anfangszeit von Drahtloskommunikation sehr großer Beliebtheit erfreut (und wird auch heute noch – in abgewandelter Form – eingesetzt).

Diese (und die nächste) Übung bauen auf der Internationalen Definition des Morse-Codes auf. Eine detaillierte Übersicht und Spezfikation desselben kann auf http://de.wikipedia.org/wiki/Morsecode gefunden werden.

Zeitschema und Codierung:

- Der Morsecode verwendet nur die drei Symbole: Punkt („dot“, oft auch als „dit“ bezeichnet), Strich („dash“) und Pause („gap“). Ein „dash“ ist dreimal so lang wie ein „dot“.
- Die Pause zwischen zwei gesendeten Symbolen ist ein „dot“ lang.
- Zwischen Buchstaben in einem Wort wird eine Pause von einem „dash“ eingeschoben.
- Die Pause zwischen Wörtern beträgt sieben „dots“.
- Es gibt keine Unterscheidung zwischen Groß- und Kleinbuchstaben.

Der nachfolgende Baum gibt an, wie Buchstaben, Ziffern und Sonderzeichen aus dem Lateinischen Alphabet in das Morsealphabet übersetzt werden können. Jeder Ast in einem linken Teilbaum entspricht einem *dot*, ein Ast in einem rechten Teilbaum entspricht einem *dash*.



Beispiele:

- „I“ = .. (*dot dot*)
- „2“ = ..--- (*dot dot dash dash dash*)
- „M“ = -- (*dash dash*)

Festlegung für die Übungen:

- Als Signalträger wird ein Audiosignal der Frequenz 2000Hz verwendet.
- Die Basis-Übertragungsrate soll 20 WPM (Words per Minute) betragen, das entspricht lt. Spezifikation einer „*dot*“-Länge von 60ms.

Aufgabenstellung für Übung 3:

In den zusammenhängenden Übungen 3 und 4 ist ein vollständiger Signalverarbeitungs-Kreis zu implementieren, wobei Übung 3 den JAVA-Teil, Übung 4 den Octave-Teil behandelt:

3A) Morse-Codierung (JAVA) (14 Punkte)

- Eingabe eines Textes (Buchstaben oder Ziffern, Sonderzeichen brauchen nicht berücksichtigt werden), frei wählbar aus einer Datei, als Kommandozeilenoption oder mittels einer GUI (Hinweis: Nur eine dieser Optionen muß implementiert werden) (2 Punkte).
- Codierung des Textes in den „Internationalen Morsecode“ (dazu ist eine „Zuordnungstabelle“ erforderlich) (6 Punkte).
- Ausgabe des Morsecodes (*„dot“*, *„dash“*, *„1-gap“*, *„3-gap*[1]*“*) frei wählbar in eine Datei, auf die Kommandozeile oder in einer GUI sowie gleichzeitige Ausgabe über die Lautsprecher als Tonfolge (entsprechend obiger Spezifikation, dh. Basislänge eines Signales „dot“, „dash“ = „3xdot“, Pause zwischen 2 Symbolen = „dot“, Pause zwischen 2 Buchstaben = „dash“ oder „3xdot“, Pause zwischen 2 Wörtern = „7xdot“). Sehen Sie die Möglichkeit vor, mit mehreren Übertragungsraten zu experimentieren (Hinweis: bei sehr kurzen „dot“-Längen, dh. hohen Übertragungsraten, kann es bei der Audio-Ausgabe zu „Buffer-Underruns“ kommen, die in weiterer Folge bei der Signaldecodierung in Übung 4 zu Problemen führen können) (6 Punkte).
- [Das Audiosignal wird in Übung 4 als Eingabe verwendet; siehe Angabe Übung 4].

3B) Morse-Decodierung (JAVA) (10 Punkte)

- [Übung 4 liefert als Ausgabe eine Datei mit dem ermittelten Morsecode. Der Dateiinhalt besteht nur aus „.“ (*dot*), „-“ (*dash*) und „ “ (*gap*); siehe Angabe Übung 4. Einige Eingabesamples zum Testen liegen im Angabeordner zur Übung].
- Öffnen und Einlesen einer Datei (bis EOF) die einen Morsecode laut folgendem Aufbau enthält (2 Punkte):
  - Ein *„dot“* ist durch einen . (Punkt) repräsentiert.
  - Ein *„dash“* ist durch einen - (Strich) repräsentiert.
  - Ein einfaches *„gap“* separiert die einzelnen Zeichen eines Wortes und wird durch ein Leerzeichen repräsentiert.
  - Trennung zwischen 2 Wörtern durch ein *„3-gap“*.
  - Dateiinhalt (Bsp.): „- .... . .-. . .. .- - ...“ („There is a Th“...)
- Dekodierung des Dateiinhalte (*„dot“*, *„dash“*, *„gap“*) in lateinische Buchstaben bzw. Ziffern (Sonderzeichen brauchen wiederum nicht berücksichtigt werden) (4 Punkte).

1-gap: Ein einzelnes „gap“ dient als Zeichen-Separator, ein „3-gap“ (3 Leerzeichen) dient als Trennzeichen zwischen je 2 Wörtern.

- Ausgabe des Klartextes, wahlweise in eine Datei, auf die Kommandozeile oder in einer GUI am Bildschirm (Hinweis: Nur eine dieser Optionen muß implementiert werden). Das Ergebnis sollte dem ursprünglich eingegebenen Text entsprechen... (4 Punkte).

Abzugeben ist ein Protokoll mit der schrittweisen Lösungsbeschreibung sowie die JAVA-Lösung (Source-Dateien), Screenshots der Applikation (beim Testen von Teil A bzw. B), einige Testfälle mit Texteingabe und ausgegenem Morsecode (Teil A) bzw. dekodierter Text der bereitgestellten Eingabesamples (Teil B).